SUPERMARIO SAYS

Systemnahe Programmierung, 2016,

Prof. Dr. Lausen

Dominik Schaufelberger, Mario Kaiser, Jasper Bröker

SUPERMARIO SAYS ...

INHALTSVERZEICHNIS

EINLEITUNG

Dieses Dokument beschreibt das Projekt "SuperMariosays ..." im Fach Systemnahe Programmierung an der DHBW Karlsruhe im Sommersemester '16 bei Professor Lausen. Ziel des Projektes soll es sein, das Spiel Senso[1] in Assembler nachzuprogrammieren. Der Code soll auf einem Simulator des *8051-Mikroprozessors*[2] und simulierter Hardware laufen. Hierzu wird die *MCU 8051 IDE*[3] genutzt.

Entwickelt wird das Projekt von den Studenten Jasper Bröker, Mario Kaiser und Dominik Schaufelberger.

SPIELPRINZIP SENSO

Senso kann allein oder mit mehreren Personen gespielt werden. Das Spiel besteht aus vier großen Feldern in den Farben Rot, Blau, Gelb und Grün. Diese leuchten abwechselnd auf und geben dabei für jede Farbe einen kurzen, individuellen Signalton von sich. Der Spieler muss sich diese Reihenfolge merken und nach Abschluss der Vorgabe durch das Spiel wiederholen. Mit jeder Runde kommt eine weitere Farb-Ton-Kombination hinzu. Mit steigendem Schwierigkeitsgrad leuchten die Felder in schnellerer Reihenfolge auf.

DIE ENTWICKLUNGSUMGEBUNG

MCU 8051 IDE

BESCHREIBUNG DER DIE

Für unser Projekt in systemnaher Programmierung nutzen wir die frei verfügbare integrierte Entwicklungsumgebung **MCU 8051 IDE,** die von Martin Osmera erstmals im Jahre 2007 in der Version 0.8 erschienen ist. Die Entwicklungsumgebung hat einen eigenen Simulator und Assembler. Außerdem unterstützt Sie die 2 Programmiersprachen C und Assembler.

Zu den Kernfunktionen der IDE zählt der Simulator an sich, der viele Debugging Funktionen, einen Interrupt Viewer, einen External Memory Viewer und einen Code Memory Viewer bietet. Außerdem besitzt der integrierte Text Editor Syntax Highlighting und Code Validierung.

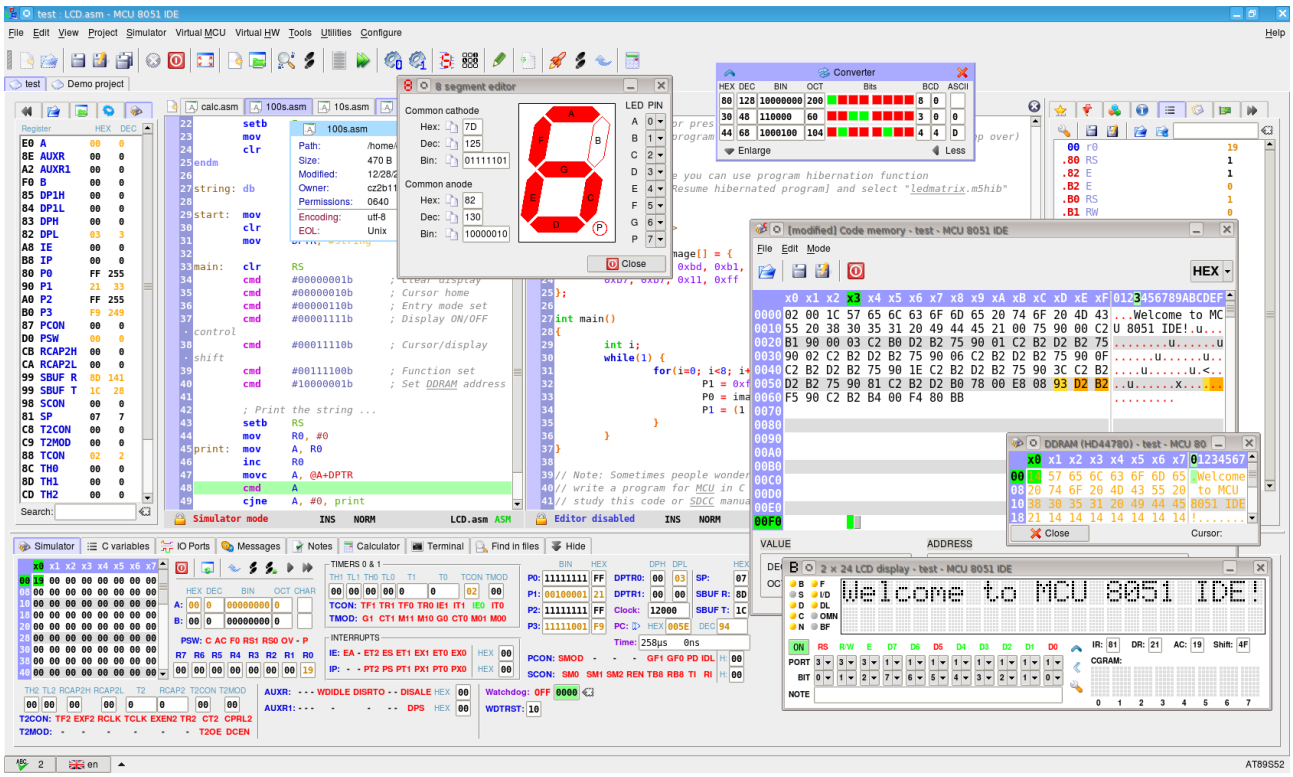
Das Bild unterhalb zeigt eine Bildschirmaufnahme von MCU 8051 IDE. Darauf sieht man, dass die IDE auch integrierte Emulatoren für LCD-Displays, 7-Segmentanzeigen und LED's hat.

---

1 https://de.wikipedia.org/wiki/Senso_(Spiel)

2 https://en.wikipedia.org/wiki/Intel_MCS-51

3 https://en.wikipedia.org/wiki/MCU_8051_IDE



EINGESETZTE SIMULIERTE HARDWARE

Damit ein Benutzer auch mit dem Programm interagieren und somit auch das Spiel effektiv spielen kann, werden zwei Komponenten der IDE benutzt. Hierbei handelt es sich um simulierte Hardware. Zum einen ein Matrix-Tastenfeld mit dem der Benutzer Eingaben tätigen kann und zum anderen eine LED Matrix, die als "Spielfeld" benutzt wird. Abbildung 1 zeigt die LED Matrix. Mit ihr wird in den nummerierten Bereichen die zufällige Reihenfolge, welche der Benutzer sich merken muss, angezeigt, indem die LEDs in diesen Bereichen zum Leuchten gebracht werden.

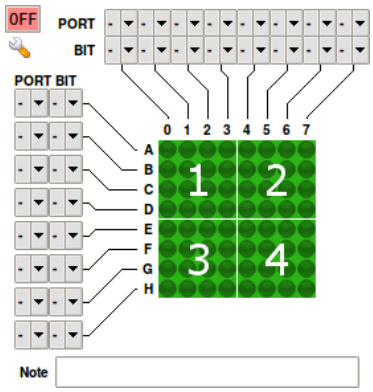


Abbildung 1 LED Matrix

Das Tastenfeld ist in Abbildung 2 zu sehen. Mit ihm kann der Benutzer die angezeigte Sequenz nachstellen um das Spiel zu gewinnen. Hierbei werden die Felder 1, 2, 4 und 5 genutzt um die Sequenz nachzustellen und A, B, C um das Spiel zu steuern. Die Felder werden wie folgt den Tasten zugeordnet:

- Feld 1 = Taste 1
- Feld 2 = Taste 2
- Feld 3 = Taste 4
- Feld 4 = Taste 5

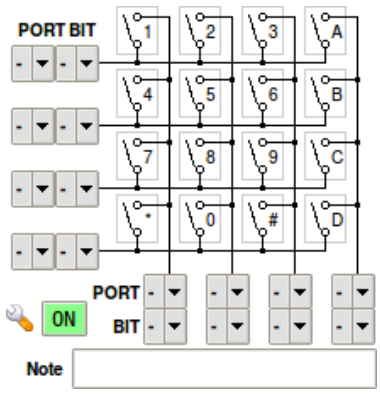


Abbildung 2 Matrix-Tastenfeld

GIT

Um die Versionierung unseres Codes und der Dokumente zu gewährleisten, benutzen wir die Versionsverwaltung von Git. Das Git Repository ist öffentlich zugänglich und kann unter folgender Adresse erreicht werden: https://github.com/1developer1/sysProg16

DAS PROGRAMM

SuperMariosays ... nutzt die bereits genannte simulierte Hardware zur Benutzerinteraktion. Über das Tastenfeld kann der Spieler das Programm starten, stoppen und neustarten. Sobald das Spiel gestartet wird wird dem Benutzer eine zufällige Folge von Feldern auf der LED Matrix angezeigt. Die Matrix ist hierbei, wie in (ABBILDUNG EINFÜGEN) zu sehen, in 4 Felder unterteilt. In einem aktiven Feld werden alle LEDs zum Leuchten gebracht. Sobald die Sequenz beendet wurde, muss der Spieler über das Keypad die zu den entsprechenden Feldern zugewiesenen Schalter drücken und die zuvor angezeigte Sequenz nachstellen.

Das Programm ist in eine Menge von Unterprogrammen unterteilt. Diese Unterpgrogramme führen bestimmte Routinen aus wie zum Beispiel das Generieren von Pseudozufallszahlen. Die Aufgaben der jeweiligen Unterprogramme werden im Folgenden erklärt.

Der interne Spielzustand wird in unterschiedlichen Bereichen des Speichers gehalten. Diese werden im folgenden tabellarisch gelistet.

Label	Beschreibung	Speicherstelle (in Hex)
rand8reg	Speichert eine generierte 8-Bit Zufallszahl.	0x20
randomAmount	Speichert die Anzahl an zu generierenden 8-Bit Zufallszahlen. Maximal können 16 Zufallszahlen generiert werden.	0x21
randomMem	Speichert die Adresse des ersten Bytes, ab welchem die zu generierenden Zufallszahlen gespeichert werden.	0x22
topLeft	Speichert die Portbelegung um das Feld 1 der LED Matrix anzusprechen.	0x50

topRight	Speichert die Portbelegung um das Feld 2 der LED Matrix anzusprechen.	0x51
bottomLeft	Speichert die Portbelegung um das Feld 3 der LED Matrix anzusprechen.	0x52
bottomRight	Speichert die Portbelegung um das Feld 4 der LED Matrix anzusprechen.	0x53
fieldMem	Speichert die Adresse des ersten Bytes, ab welchem die zu den Zufallszahlen gehörigen Felder gespeichert werden. Jeder Zufallszahl wird dabei ein fester Wert zugeordnet. Dieser wird benutzt um die Benutzereingabe zu validieren. Diese Speicherstelle wird mit dem Wert $\#60H$ initialisiert. Der letzte mögliche Wert der Sequenz kann demnach an der Stelle $60H + randomAmount = \#6F$ stehen.	0x54
sequenceMem	Speichert die Adresse des ersten Bytes, ab welchem die zu den Zufallszahlen Portbelegungen gespeichert werden. Jeder Zufallszahl wird dabei eine bestimmte Portbelegung (aus topLeft, topRight, bottomLeft, bottomRight) zugewiesen. Diese Speicherstelle wird mit dem Wert $\#70H$ initialisiert. Die letzte mögliche Portbelegung der Sequenz kann demnach bei $\#70H + randomAmount = \#7F$ stehen.	0x55

UNTERPROGRAMME

HAUPTSPIELSCHLEIFE

Die Hauptspielschleife steuert den kompletten Ablauf des Programms und ruft die jeweiligen anderen Unterprogramme auf. Wenn das Programm gestartet wird, wird gleichzeitig ein Timer des Mikrocontrollers gestartet. Über die *Start / Stop / Neustart* Routine kann dann das Spiel begonnen werden. Wenn das Spiel gestartet wurde, wird der *Feldsequenz-Generator* aufgerufen. Dieser erzeugt eine zufällige Reihenfolge für das ansprechen der Felder. Nachdem die Sequenz vollständig generiert wurde, wird das Unterprogramm *Feldsequenz Visualisieren* ausgeführt und visualisiert dem Spieler die Sequenzreihenfolge. Wenn die Anzeige der Sequenz beendet ist, wird die Benutzereingabe erwartet. Diese wird dann auf Korrektheit geprüft. Der Spieler bekommt dann Rückmeldung ob seine eingegebene Sequenz korrekt war oder nicht.

START / STOP / NEUSTART

Die Start / Stop / Neustart Routine ist für den Spielablauf zuständig. Mit dieser wird ein neues Spiel gestartet, ein Spiel beendet oder ein Spiel neugestartet.

Start:
Wenn die Start-Routine aufgerufen wird, muss die Hauptschleife gestartet werden.

Stop:
Wenn die Stop-Routine aufgerufen wird, muss die Hauptschleife beendet werden.

Neustart:
Wenn die Neustart-Routine aufgerufen wird, muss zunächst die Stop-Routine aufgerufen werden und im Anschluss die Start-Routine. So kann mit schon vorhandenen Mitteln ein neues Spiel gestartet werden.

PSEUDOZUFALLSZAHLEN GENERATOR

Wenn der Pseudozufallszahlen Generator aufgerufen wird, dann wird zuerst der aktuelle Wert des Timers abgerufen. Dieser dient als Startwert für die Berechnung der Pseudozufallszahlen anhand verschiedener Operationen. Die erzeugten Zufallszahlen werden dann im Speicher abgelegt um sie später für den *Feldsequenz-Generator* zu verwenden.

FELDSEQUENZ-GENERATOR

Der Feldsequenz-Generator erzeugt aus den generierten Pseudozufallszahlen die für die Visualisierung benötigten Portbelegungen. Aus einer Pseudozufallszahl wird über eine Modulo-Operation der Index eines Feldes berechnet. Für jeden eindeutigen Index wird eine bestimmte eindeutige Portbelegung erzeugt. Diese Portbelegung wird wieder im Speicher abgelegt um diese in der *Feldsequenz Visualisierung* zu nutzen.

FELDSEQUENZ VISUALISIEREN

Das Unterprogramm zur Visualisierung

Jasper

BENUTZEREINGABE

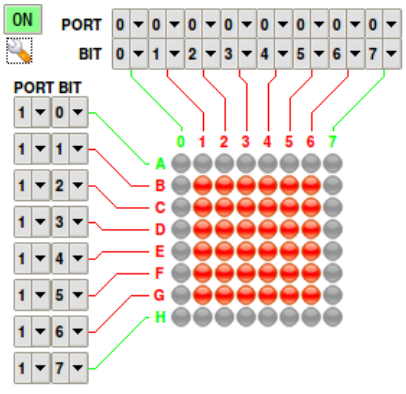
Mario

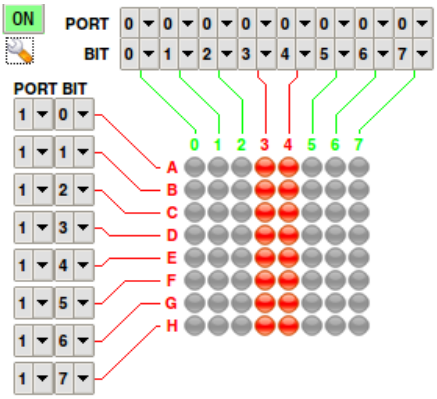
VERGLEICHEN DER BENUTZEREINGABE MIT DER SEQUENZ

Mario

KORREKTHEITS-ANZEIGE

Die Korrektheits-Anzeige wird abhängig vom Ergebnis des Vergleiches der Benutzereingabe mit der Sequenz gesteuert. Sollte der Benutzer die Sequenz richtig wiedergegeben haben, dann leuchtet für einen kurzen Augenblick, eine „0“ auf der LED-Matrix auf. Für den Fall, dass der Benutzer eine nicht korrekte Eingabe getätigt haben sollte, eine „1“ auf der LED-Matrix ausgegeben.





IMPLEMENTIERUNG

BENUTZERTEST

FAZIT